BDI & INTRODUCTION TO JASON

BE4M36MAS - Multiagent systems

BELIEF-DESIRE-INTENTION

What is it?

Model for programming autonomous agents using three concepts:

- **Beliefs**
- **Desires**
- **Intentions**

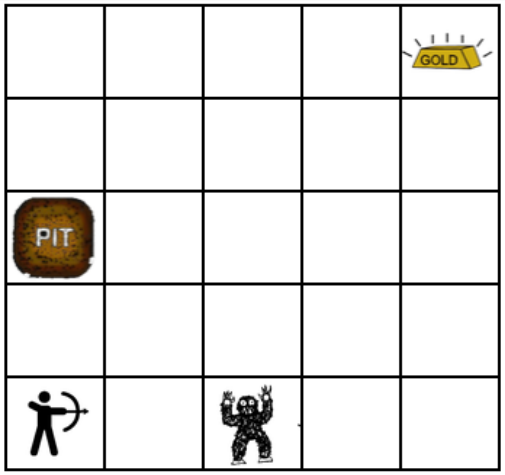
Beliefs

Beliefs

$\sim$ agent's model of the world (what he supposes to be true)

Example:

```
breeze(0, 1).  stench(1, 0).
pos(0, 0).  safe(0, 0).
safe(0, 1).  safe(1, 0).
```



Beliefs

Beliefs **are not** knowledge!

- An agent may **believe** facts that are **not true**.

Example:

Weather forecast announces nice weather for the weekend.

```
nice_weather(sat).  nice_weather(sun).
```

→ You can believe that, but you cannot take it for granted.

Desires

Desires

$\sim$ state of the world agent is **dreaming** about

Agent need not succeed in achieving all his desires, e.g.:
$\rightarrow$ Situation may not allow completing some of the desires
$\rightarrow$ Desires may be mutually exclusive

Intentions

Intentions

$\sim$ **Active** goals of the agent (should **not contradict** beliefs)

Agent **commits** to fulfiling some of his desires. He must do everything he can to complete his intentions (unless specified otherwise).

Turning desires into intentions: **deliberation**
Realizing intentions: **means-ends reasoning**

Example:

```
!organize_picnic(sat).
```

Commitments

Commitments

$\sim$ indicate that an agent has **commited** to some intention

Optional: Situation in which an agent may forget about his intention (i.e. **decommit**)

- Individual commitments
- Social commitments

Individual commitments

- **Blind commitment** — the only way to decommit is to succeed
- **Single-minded commitment** — agent may decommit when he believes it is no longer possible to succeed
- **Open-minded commitment** — agent may decommit when he no longer believes it is possible to succeed

Individual commitments

Example:

Recall `!organize_picnic(sat).` and `nice_weather(sat).`

Picnic can be organized only in good weather conditions.

Individual commitments

Example:

Recall `!organize_picnic(sat).` and `nice_weather(sat).`
Picnic can be organized only in good weather conditions.

- **Blind commitment**

Individual commitments

Example:

Recall `!organize_picnic(sat).` and `nice_weather(sat).`
Picnic can be organized only in good weather conditions.

- **Blind commitment** — Agent will be organizing the picnic event. Once he realizes that it's raining whole Saturday, he crashes.

Individual commitments

Example:

Recall `!organize_picnic(sat).` and `nice_weather(sat).`
Picnic can be organized only in good weather conditions.

- **Blind commitment** — Agent will be organizing the picnic event. Once he realizes that it's raining whole Saturday, he crashes.
- **Single-minded commitment**

Individual commitments

Example:

Recall `!organize_picnic(sat).` and `nice_weather(sat).`
Picnic can be organized only in good weather conditions.

- **Blind commitment** — Agent will be organizing the picnic event. Once he realizes that it's raining whole Saturday, he crashes.
- **Single-minded commitment** — Agent will be organizing the event until rainy Saturday. He than resigns on his intention and the life goes by.

Individual commitments

Example:

Recall `!organize_picnic(sat).` and `nice_weather(sat).`
Picnic can be organized only in good weather conditions.

- **Blind commitment** — Agent will be organizing the picnic event. Once he realizes that it's raining whole Saturday, he crashes.
- **Single-minded commitment** — Agent will be organizing the event until rainy Saturday. He than resigns on his intention and the life goes by.
- **Open-minded commitment**

Individual commitments

Example:

Recall `!organize_picnic(sat).` and `nice_weather(sat).`
Picnic can be organized only in good weather conditions.

- **Blind commitment** — Agent will be organizing the picnic event. Once he realizes that it's raining whole Saturday, he crashes.
- **Single-minded commitment** — Agent will be organizing the event until rainy Saturday. He than resigns on his intention and the life goes by.
- **Open-minded commitment** — Agent drops his intention as soon as the updated forecast is released.

JASON PROGRAMMING

Key components

- Set of beliefs (**belief base**)
- Set of intentions

Key components

- Set of beliefs (**belief base**)
- Set of intentions

- Plan library
- Set of events

Specifying initial beliefs and intentions

- **Specifying beliefs**

  $\sim$ what agent knows at the beginning of the execution

```
depot(5,5).
next(X,X+1).
```

- **Specifying intentions**

  $\sim$ what agent has to accomplish

```
!say_hello.
!find_gold.
!go_to(5,5).
```

Building plan database

```
!say_hello.
```

Building plan database

```
!say_hello.
```

In order to execute an intention there must be an appropriate **plan**:

```
+!say_hello <- .print("Hello").
```

Building plan database

Incorporating beliefs:

```
daytime(afternoon).
!say_hello.
```

Building plan database

Incorporating beliefs:

```
daytime(afternoon).
!say_hello.
```

```
  +!say_hello :  daytime(morning) <- .print("Good morning").
+!say_hello :  daytime(afternoon) <- .print("Good afternoon").
  +!say_hello :  daytime(evening) <- .print("Good evening").
```

Building plan database

Incorporating variables (starting with **capital** letters):

```
daytime(afternoon).
!say_hello("Bob").
```

```
+!say_hello(X) : daytime(T) <- .print("Good ", T, ", ", X).
```

Building plan database

$$\underbrace{+!\texttt{say\_hello(X)}}_{\text{trigger}} : \underbrace{\texttt{daytime(T)}}_{\text{context}} \texttt{ <- } \underbrace{\texttt{.print("Good ",T,",",X)}}_{\text{plan / subgoals}} .$$

- **Trigger** — what event does the plan handle
- **Context** — condition when the plan is applicable
- **Plan** — what has to be done in order to fulfil the intention

Variables get **unified**, i.e. they get the value matching the intention specification / belief base.

Building plan database

- **Triggers**

	+ (additions)	- (removals)
Intentions	+!*intention*(*args*)	-!*intention*(*args*)
Beliefs	+*belief*(*args*)	-*belief*(*args*)

- **Context** — Logical formula (using beliefs and percepts)
  `a & b`, `a | b`, `not a` (belief `a` is not present), `~a`
  Percepts: `pos(X,Y)`, `cell(X,Y,T)` (`T=gold,depot,ally`), `carrying_gold`, `name(N)`, `gsize(_,W,H)`
- **Plan** — subgoals to achieve (separated by `;`)
  - Intentions — `!subgoal` (sequential) / `!!subgoal`
  - Environment actions — `do(left)`, `do(right)`, `do(up)`, `do(down)`, `do(pick)`, `do(drop)`
  - Internal actions — e.g. `.print("Hello")`
  - Belief base manipulation — `+belief`, `-belief`

EXAMPLE

Example: Call forwarding rules

Alice:

- During lunchtime, forward all calls to Carla.
- When I am busy, incoming calls from colleagues should be forwarded to Denise.

Example: Call forwarding rules

- During lunchtime, forward all calls to Carla.
  - What is lunchtime? — belief `lunchtime(1130, 1230).`
  - What is the incoming call? — addition of belief `invite(X,Y)`

Example: Call forwarding rules

- During lunchtime, forward all calls to Carla.
  - What is lunchtime? — belief `lunchtime(1130, 1230).`
  - What is the incoming call? — addition of belief `invite(X,Y)`
  - So...

```
+invite(X,alice) :  time(T) & lunchtime(S,E) &
S<=T & T<=E <- !call_forward(alice, X, carla).
```

Example: Call forwarding rules

- When I am busy, incoming calls from colleagues should be forwarded to Denise.
  - What it means to be busy? — percept `status(X,busy)`
  - Who is a colleague? — belief `colleague(X)`

Example: Call forwarding rules

- When I am busy, incoming calls from colleagues should be forwarded to Denise.
  - What it means to be busy? — percept `status(X,busy)`
  - Who is a colleague? — belief `colleague(X)`
  - So...

```
+invite(X,alice) :
status(alice,busy) & colleague(X) <-
  !call_forward(alice, X, denise).
```

Example: Call forwarding rules

- Nothing prevents the connection:
  - How to make the connection? — Environment action `connect(X,Y)`

Example: Call forwarding rules

- Nothing prevents the connection:
  - How to make the connection? — Environment action `connect(X,Y)`
  - So...

```
+invite(X,Y) : status(Y,idle) <- connect(X,Y).
```

Example: Call forwarding rules

- How to achieve the call forwarding?

Example: Call forwarding rules

- How to achieve the call forwarding?

```
+!call_forward(X,Y,Z) : status(Z,idle) <-
        -invite(X,Y) ; +invite(X,Z).
```